



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/827,974	04/06/2001	Jason Souloglou		7224

23483 7590 06/19/2006

WILMER CUTLER PICKERING HALE AND DORR LLP
60 STATE STREET
BOSTON, MA 02109

EXAMINER

YIGDALL, MICHAEL J

ART UNIT	PAPER NUMBER
----------	--------------

2192

DATE MAILED: 06/19/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.

09/827,974

Applicant(s)

SOULOGLOU ET AL.

Examiner

Michael J. Yigdall

Art Unit

2192

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 10 March 2006.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-30 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-30 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. This Office action is responsive to Applicant's submission filed on March 10, 2006.

Claims 1-30 are pending.

Response to Arguments

2. Applicant's arguments have been considered but are moot in view of the new ground(s) of rejection, as set forth below.

Terminal Disclaimer

3. The terminal disclaimer filed on March 10, 2006 disclaiming the terminal portion of any patent granted on this application that would extend beyond the expiration date of any patent granted on Application No. 10/164,789 has been reviewed and is accepted. The terminal disclaimer has been recorded.

4. The terminal disclaimer filed on March 10, 2006 disclaiming the terminal portion of any patent granted on this application that would extend beyond the expiration date of any patent granted on Application No. 10/165,457 has been reviewed and is accepted. The terminal disclaimer has been recorded.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an

Art Unit: 2192

international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

6. Claims 13, 19, 24 and 28 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 6,427,234 to Chambers et al. (now made of record, "Chambers").

With respect to claim 13 (currently amended), Chambers discloses a method of generating a target code representation of program code (see, for example, column 9, lines 15-27), the method comprising the computer implemented steps of:

on an initial translation of a given portion of the program code, generating and storing only target code which is required to execute that portion of program code with a prevailing set of conditions (see, for example, column 9, lines 33-43, which shows generating and storing only the target code needed to execute a unit under a prevailing context); and

whenever subsequently the same portion of the program code is entered, determining whether the prevailing set of conditions are now different, subsequent conditions, and if no target code compatible with the subsequent conditions has previously been generated, generating additional target code required to execute said portion of program code with said subsequent conditions (see, for example, column 9, lines 33-43, which shows determining whether the context is different when the unit is subsequently entered, and if target code for the subsequent context was not already generated, generating the additional target code needed to execute the unit under the subsequent context).

With respect to claim 19 (previously presented), the rejection of claim 13 is incorporated, and Chambers further discloses that said target code representation is generated at run time (see, for example, column 9, lines 15-27).

With respect to claim 24 (previously presented), the rejection of claim 13 is incorporated, and Chambers further discloses that said steps are performed at run time (see, for example, column 9, lines 15-27).

With respect to claim 28 (previously presented), the rejection of claim 13 is incorporated, and Chambers further discloses that whenever subsequently the same portion of program code is entered, the method comprises the computer implemented steps of:

determining whether said subsequent conditions are different from said prevailing set of conditions (see, for example, column 9, lines 33-43, which shows determining whether the subsequent context is different than the prevailing context for which target code was already generated and stored), and

if different, generating and storing different target code for said same portion of program code required to execute said portion of program code with said subsequent conditions (see, for example, column 9, lines 33-43, which shows generating and storing different target code for the unit to execute under the subsequent context if the contexts are different).

7. Claim 14 is rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Patent No. 5,930,509 to Yates et al. (now made of record, "Yates").

With respect to claim 14 (previously presented), Yates discloses a method of dynamically translating first computer program code written for a first programmable machine into second computer program code for running on a different second programmable machine (see, for example, column 2, lines 3-18, and column 7, lines 35-49, which further shows translating first, non-native computer program code into second, native computer program code), said method comprising:

(a) generating an intermediate representation of a block of said first computer program code (see, for example, column 70, lines 7-21, which shows generating an intermediate representation of the first computer program code);

(b) generating a block of said second computer program code from said intermediate representation (see, for example, column 70, lines 22-29, which shows generating the second computer program code from the intermediate representation);

(c) running said block of second computer program code on said second programmable machine (see, for example, column 9, lines 19-34, which shows executing the second computer program code); and

(d) repeating steps (a)-(c) in real time for at least the blocks of first computer program code needed for a current emulated execution of the first computer program code on said second programmable machine (see, for example, column 8, lines 26-35, which shows repeating the translation steps at run-time).

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

Art Unit: 2192

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

9. Claims 1-12 and 15-18, 20-23, 25-27, 29 and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yates in view of Chambers.

With respect to claim 1 (currently amended), Yates discloses a method of generating an intermediate representation of program code (see, for example, column 2, lines 3-18, and column 73, lines 26-41, which further shows generating an intermediate representation), the method comprising the computer implemented steps of:

on an initial translation of a given portion of program code, generating and storing only intermediate representation which is required to execute that portion of program code with a prevailing set of conditions (see, for example, column 70, lines 7-21, which shows generating and storing an intermediate representation of a translation unit of the program code).

whenever subsequently the same portion of the program code is entered, if no intermediate representation compatible with the subsequent conditions has previously been generated, generating additional intermediate representation required to execute said portion of program code with said subsequent conditions (see, for example, column 9, lines 19-34, which shows translating additional program code and thus generating an additional intermediate representation of the program code if it was not already generated).

Yates discloses that run-time profile information directs the translation of program code (see, for example, column 9, lines 39-52), but does not expressly disclose:

whenever subsequently the same portion of the program code is entered, determining whether the prevailing set of conditions are now different, subsequent conditions.

However, Chambers discloses a method of generating specialized or special-case program code based on run-time conditions (see, for example, column 9, lines 15-27). The method comprises generating and storing program code for a unit under a prevailing context, and whenever the unit is subsequently entered, determining whether the prevailing context is now a different, subsequent context, and generating additional program code for the unit under the subsequent context if it was not already generated (see, for example, column 9, lines 33-43). Chambers discloses that specialization enhances program performance (see, for example, column 7, lines 17-20), such as with optimizations based on values computed only at run-time (see, for example, column 1, lines 42-46).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Yates with the specialization features of Chambers, such that Yates generates and stores intermediate representations for only the conditions prevailing at the time the portion of the program code is entered, and generates additional intermediate representations as needed for different, subsequent conditions. One of ordinary skill in the art at the time the invention was made would have been motivated to supplement Yates with the optimizations that specialization provides, so as to enhance program performance. In fact, the method of Chambers applies to the problem of portability across different hardware architectures (see, for example, Chambers, column 1, lines 63-67), which is of course a concern of Yates (see, for example, Yates, column 6, lines 56-62).

With respect to claim 2 (original), the rejection of claim 1 is incorporated, and Yates in view of Chambers further discloses that the conditions are entry conditions (see, for example, Chambers, column 7, lines 50-54, which shows that the conditions are entry conditions), and the method comprises the computer implemented steps of:

generating an Intermediate Representation Block (IR Block) of intermediate representation for each Basic Block of program code as it is required by the program, each IR Block representing a respective Basic Block of program code for a particular entry condition (see, for example, Yates, column 73, lines 26-41, which shows generating such an IR block);

storing target code corresponding to each IR Block (see, for example, Yates, column 70, lines 22-29, which shows generating and storing such target code); and

when the program requires execution of a Basic Block for a given entry condition, either:

(a) if there is stored target code representing that Basic Block for that given entry condition, using said stored target code (see, for example, Chambers, column 9, lines 33-43, which shows using stored target code if it exists for a given entry condition); or

(b) if there is no stored target code representing that Basic Block for that given entry condition, generating a further IR Block representative of that Basic Block for that given entry condition (see, for example, Chambers, column 9, lines 33-43, which shows generating further target code if it does not exist for the given entry condition, and Yates, column 57, lines 1-8, which shows generating an IR block to generate such target code).

With respect to claim 3 (original), the rejection of claim 1 is incorporated, and Yates in view of Chambers further discloses that the intermediate representation of the program code is generated dynamically as the program code is running (see, for example, Yates, column 8, lines

Art Unit: 2192

26-35, which shows that the translation and thus the generation of the intermediate representation is done at run-time), the method comprising the computer implemented steps of:

at a first iteration of a particular subject code instruction having a plurality of possible effects or functions, generating and storing special-case intermediate representation representing only the specific functionality required at that iteration (see, for example, Chambers, column 9, lines 33-43, which shows generating specialized or special-case code for a unit representing only the functionality required at that iteration, and Yates, column 57, lines 1-8, which shows generating an intermediate representation to generate such code); and

at each subsequent iteration of the same subject code instruction, determining whether special-case intermediate representation has been generated for the required functionality required at said subsequent iteration and generating additional special-case intermediate representation specific to that functionality if no such special-case intermediate representation has previously been generated (see, for example, Chambers, column 9, lines 33-43, which shows generating additional specialized or special-case code for a unit representing only the functionality required at that iteration if it was not already generated, and Yates, column 57, lines 1-8, which shows generating an intermediate representation to generate such code).

With respect to claim 4 (previously presented), the rejection of claim 3 is incorporated, and Yates in view of Chambers further discloses that said special-case intermediate representation is generated and stored and an associated test procedure is generated and stored to determine on subsequent iterations of the respective subject code instruction whether the required functionality is the same as that represented by the associated stored special-case intermediate representation, and where additional special-case intermediate representation is

required an additional test procedure associated with that special-case intermediate representation is generated and stored with that additional special-case intermediate representation (see, for example, Chambers, column 33, lines 49-59, which shows generating and storing such test procedures).

With respect to claim 5 (previously presented), the rejection of claim 4 is incorporated, and Yates in view of Chamber further discloses that the additional special case intermediate representation for a particular subject code instruction and the additional associated test procedure is stored at least initially in subordinate relation to any existing special-case intermediate representation and associated test procedures stored to represent the same subject instruction, such that upon the second and subsequent iteration of a subject code instruction, a determination of whether or not required special-case intermediate representation has previously been generated is made by performing said test procedures in the order in which they were generated and stored until either it is determined that special-case intermediate representation of the required functionality exists, or it is determined that no such required special-case intermediate representation exists in which case more additional intermediate representation and another associated test procedure is generated (see, for example, Chambers, column 30, lines 37-54, which shows that the specialized or special-case code and the test procedures are generated and stored in order).

With respect to claim 6 (previously presented), the rejection of claim 5 is incorporated, and Yates in view of Chambers further discloses that the intermediate representation is optimised by adjusting the ordering of the test procedures such that a test procedure associated with a more

Art Unit: 2192

frequently used special-case intermediate representation is run before a test procedure associated with a less frequently used special-case intermediate representation rather than ordering the test procedures in the order in which they are generated (see, for example, Yates, column 71, lines 13-25, which shows such an optimization).

With respect to claim 7 (original), the rejection of claim 1 is incorporated, and Yates in view of Chambers further discloses translating the program code written for execution by a processor of a first type so that the program code may be executed by a processor of a second type, using the generated intermediate representation (see, for example, Yates, column 7, lines 35-49, which shows translating non-native program code into native program code).

With respect to claim 8 (original), the rejection of claim 7 is incorporated, and Yates in view of Chambers further discloses that said translation is dynamic and performed as the program code is run (see, for example, Yates, column 8, lines 26-35, which shows that the translation is done at run-time).

With respect to claim 9 (original), the rejection of claim 1 is incorporated, and Yates in view of Chambers further discloses optimising the program code by optimising said intermediate representation (see, for example, Yates, column 74, lines 46-54, which shows optimizing the intermediate representation).

With respect to claim 10 (original), the rejection of claim 9 is incorporated, and Yates in view of Chambers further discloses that the method is used to optimise the program code written for execution by a processor of a first type so that the program code may be executed more

efficiently by that processor (see, for example, Yates, column 46, lines 20-30, which shows optimizing the program code for more efficient execution).

With respect to claim 11 (original), Yates discloses a method for generating an intermediate representation of program code written for running on a programmable machine (see, for example, column 2, lines 3-18, and column 73, lines 26-41, which further shows generating an intermediate representation), said method comprising:

(i) generating a plurality of register objects for holding variable values to be generated by the program code (see, for example, column 73, lines 42-55, which shows generating such register objects); and

(ii) generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code (see, for example, column 60, lines 5-25, which shows generating such expression objects);

said intermediate representation being generated and stored for a block of program code and subsequently re-used if the same block of program code is later re-entered (see, for example, column 9, lines 19-34, which shows reusing the translated program code and thus the intermediate representation from which it was generated if the same program code is later reentered).

Yates does not expressly disclose that at least one block of program code can have alternative un-used entry conditions or effects or functions and said intermediate representation is only initially generated and stored as required to execute that block of program code with a then prevailing set of conditions.

However, Chambers discloses a method of generating specialized or special-case program code based on run-time conditions (see, for example, column 9, lines 15-27). The method comprises generating and storing program code for a unit under a prevailing context, wherein the unit can have other, unused contexts (see, for example, column 9, lines 33-43). The context comprises entry conditions (see, for example, column 7, lines 50-54). Chambers discloses that specialization enhances program performance (see, for example, column 7, lines 17-20), such as with optimizations based on values computed only at run-time (see, for example, column 1, lines 42-46).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement Yates with the specialization features of Chambers, such that Yates generates and stores an intermediate representation for only the conditions prevailing at the time the block of program code is entered. One of ordinary skill in the art at the time the invention was made would have been motivated to supplement Yates with the optimizations that specialization provides, so as to enhance program performance. In fact, the method of Chambers applies to the problem of portability across different hardware architectures (see, for example, Chambers, column 1, lines 63-67), which is of course a concern of Yates (see, for example, Yates, column 6, lines 56-62).

With respect to claim 12 (original), the rejection of claim 11 is incorporated, and Yates in view of Chambers further discloses that for a given block of program code, it is determined whether a previously stored intermediate representation therefor was for the same now currently prevailing set of conditions and, if not, then generating and storing additional intermediate representation as required to execute the block of program code for the new now currently

Art Unit: 2192

prevailing set of conditions (see, for example, Yates, column 9, lines 19-34, which shows translating additional program code and thus generating an additional intermediate representation of the program code if it was not already generated).

With respect to claim 15 (currently amended), the claim recites a system that is analogous to the method of claim 1 (see the rejection of claim 1 above).

With respect to claim 16 (original), the claim recites a system that is analogous to the method of claim 11 (see the rejection of claim 11 above).

With respect to claim 17 (previously presented), the rejection of claim 1 is incorporated, and Yates in view of Chambers further discloses that said intermediate representation of program code is generated at run time (see, for example, Yates, column 8, lines 26-35, which shows that the translation and thus the generation of the intermediate representation is done at run-time).

With respect to claim 18 (previously presented), the rejection of claim 1 is incorporated, and Yates in view of Chambers further discloses that said intermediate representation of program code is generated at run time (see, for example, Yates, column 8, lines 26-35, which shows that the translation and thus the generation of the intermediate representation is done at run-time).

With respect to claim 20 (previously presented), the rejection of claim 15 is incorporated, and Yates in view of Chambers further discloses that said intermediate representation of program code is generated at run time (see, for example, Yates, column 8, lines 26-35, which shows that the translation and thus the generation of the intermediate representation is done at run-time).

With respect to claim 21 (previously presented), the rejection of claim 16 is incorporated, and Yates in view of Chambers further discloses that said intermediate representation of program code is generated at run time (see, for example, Yates, column 8, lines 26-35, which shows that the translation and thus the generation of the intermediate representation is done at run-time).

With respect to claim 22 (previously presented), the rejection of claim 1 is incorporated, and Yates in view of Chambers further discloses that said steps are performed at run time (see, for example, Yates, column 8, lines 26-35, which shows that the steps are performed at run-time).

With respect to claim 23 (previously presented), the rejection of claim 11 is incorporated, and Yates in view of Chambers further discloses that said plurality of register objects and plurality of expression objects are generated at run time (see, for example, Yates, column 8, lines 26-35, which shows that the translation and thus the generation of the intermediate representation is done at run-time).

With respect to claim 25 (previously presented), the rejection of claim 15 is incorporated, and Yates in view of Chambers further discloses that the function of generating and storing on an initial translation, the function of determining, and the function of generating additional intermediate representation are each performed at run time (see, for example, Yates, column 8, lines 26-35, which shows that the functions are performed at run-time).

With respect to claim 26 (previously presented), the rejection of claim 16 is incorporated, and Yates in view of Chambers further discloses that the functions of generating a plurality of

Art Unit: 2192

register objects, generating a plurality of expression objects, and generating and storing intermediate representation are each performed at run time (see, for example, Yates, column 8, lines 26-35, which shows that the functions are performed at run-time).

With respect to claim 27 (previously presented), the rejection of claim 1 is incorporated, and Yates in view of Chambers further discloses that whenever subsequently the same portion of program code is entered, the method comprises the computer implemented steps of:

determining whether said subsequent conditions are different from said prevailing set of conditions (see, for example, Chambers, column 9, lines 33-43, which shows determining whether the subsequent context is different than the prevailing context), and

if different, generating and storing a different intermediate representation for said same portion of program code required to execute said portion of program code with said subsequent conditions (see, for example, Chambers, column 9, lines 33-43, which shows generating different target code for the unit under the subsequent context if the contexts are different, and Yates, column 57, lines 1-8, which shows generating an intermediate representation to generate such target code).

With respect to claim 29 (previously presented), the rejection of claim 11 is incorporated, and Yates in view of Chambers further discloses that said at least one block of program code has a plurality of possible alternative entry conditions or effects or functions (see, for example, Chambers, column 7, lines 50-54, which shows that the unit has a plurality of possible alternative entry conditions),

further wherein said intermediate representation is only initially generated and stored to represent only a portion of said plurality of possible alternative entry conditions or effects or functions as required to execute that block of program code with said prevailing set of conditions (see, for example, Chambers, column 9, lines 33-43, which shows generating only the target code needed to execute the unit under the prevailing entry conditions, and Yates, column 57, lines 1-8, which shows generating an intermediate representation to generate such target code).

With respect to claim 30 (previously presented), the rejection of claim 29 is incorporated, and Yates in view of Chambers further discloses that said intermediate representation does not represent any un-used entry conditions or effects or functions from said plurality of possible alternative entry conditions or effects or functions (see, for example, Chambers, column 9, lines 33-43, which shows that any unused contexts are not represented).

Conclusion

10. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure. U.S. Patent No. 6,463,582 to Lethin et al. discloses a dynamic optimizing object code translator for architecture emulation and a dynamic optimizing object code translation method.

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

Art Unit: 2192

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy


TUAN DAM
SUPERVISORY PATENT EXAMINER